JOURNAL OF APPLIED INTELLIGENT SYSTEMS & INFORMATION SCIENCES

VOL. 5, ISSUE 1, PP. 1-16, FEBRUARY 2024.
Available at: **https://journal.research.fanap.com/**
DOI: https://doi.org/10.22034/JAISIS.2024.408658.1063

THE IMPACT OF AGILE SOFTWARE DEVELOPMENT ON BANKING SOFTWARE PROVIDERS: A CASE STUDY

Maryam Yahyaie [1], Mohammad Rabiei [1,*]

[1] *Datis Arian Qeshm (Dotin), Tehran, Iran.*

ABSTRACT

Banking software development is a complex and dynamic field that requires constant adaptation to changing customer requirements and market conditions. Traditional software development methods, such as Waterfall, may not be suitable for such a context, as they are rigid, sequential, and slow. Agile Software Development (ASD) is an alternative approach that emphasizes flexibility, collaboration, and customer satisfaction. This paper presents a case study of a banking software company that adopted Scrum, to improve the software development process and customer satisfaction. The paper describes the executive procedure of the production cycle, which consists of three phases: seasonal planning, production, and delivery. The paper also reports the key performance indicators (KPIs) used to measure the impact of Scrum on software quality and delivery. The paper discusses the challenges and opportunities of using Scrum, such as the need for customer involvement, the difficulty of integration, and the lack of systematic requirements management. The paper also analyzes the cultural and organizational factors and the difficulties of coordination among heterogeneous software development teams. The paper concludes that Scrum has brought several benefits to the company, such as shorter delivery timeframes, improved communication, increased team participation, early identification of problems, and system improvement. However, the paper also acknowledges some limitations and challenges of Scrum. It contributes to the literature on ASD by providing a comprehensive and in-depth case study of Scrum adoption in a banking software company. It also provides insights and recommendations for practitioners and researchers who are interested in applying ASD in similar contexts.

KEYWORDS: Agile Software Development, Banking Software Development, Organizational Capability.

1. INTRODUCTION

Software development has been a dynamic and diverse field, with many models, methods, and standards proposed over the years. However, there is no consensus on which software development approach is the best for every situation. Software development settings vary depending on factors such as the type of application, the size of the team, the volatility of the requirements, and the experience of the personnel (Clarke et al., 2015). Moreover, these settings are not static but change over time (O'Connor & Clarke, 2015). Therefore, software processes need to be adapted and tailored to fit the specific context of each project (Coleman & O’Connor, 2008).

[*] *Corresponding Author, Email: mohammad.rabiei@dotin.ir*

RECEIVED: 25 JULY 2023; **ACCEPTED**: 09 JANUARY 2024; **PUBLISHED ONLINE**: 28 FEBRUARY 2024

Software processes are not fixed, but continuous and evolving (Curtis, 1989). Thus, it is important to understand how software processes interact with their situational contexts and how they can be improved accordingly (Clarke & O'Connor, 2012). The optimal software development process depends on the characteristics of each setting. In the current competitive market, software development organizations face huge pressure to deliver software-intensive systems faster and more frequently. Software releases that used to happen once or twice a year have now become weekly, daily, or even hourly. This requires organizations to innovate and release software in shorter and parallel cycles, which involves adopting new practices in the industry.

Agile Software Development (ASD) is becoming more popular in the tech sector, as it offers more flexibility than traditional plan-driven modeling techniques (Abrahamsson et al., 2006; Chapman & White, 2016; Chapman et al., 2017; Glas & Ziemer, 2009; Paige et al., 2011). A 2018 survey of technology professionals found that 97 percent of them use agile approaches. Moreover, the survey reported that 78 percent of the respondents said that their organizations use a hybrid of agile and plan-based methods and practices. The advocates of Agile Software Development argue that planned software applications cannot cope with fast-changing business needs (Andres, 2005; Hohl et al., 2018). ASD addresses this need for adaptability by emphasizing small co-located teams, short development cycles, and frequent informal communication among software team members while avoiding activities that do not add value to the project customer, such as documentation (Black et al., 2009). Some examples of agile approaches that embody these principles are Feature Driven Development (Felsing & Palmer, 2002), Extreme Programming (XP) (Andres, 2005), and Scrum. Each agile approach has its techniques, such as daily standup in Scrum or pair programming in XP. These techniques can be modified by adding or removing practices, or by customizing the practices themselves, to suit the project environment. Several researchers have suggested that Agile Software Development is more suitable for smaller projects (Boehm, 2002; Boehm et al., 2004). There is evidence that ASD works well in these situations (Paetsch et al., 2003).

Banking software development companies (BSDCs) are specialized in creating software and systems for the banking industry. Most of these companies work for a single bank, but some have clients from different banks. The ASD process needs to consider various issues and situations in a development environment (McLeod & MacDonell, 2011; Sengupta et al., 2006). This contextualization allows a better understanding of who, where, when, and why something works (Fruhling & Vreede, 2006). This change affects how products are designed, developed, tested, and delivered to users.

The software process is the term used to describe the actions, techniques, practices, and technologies that people and organizations use to create and maintain software and related products (McLeod & MacDonell, 2011). The interest in the software process is based on two key assumptions, (1) The quality of the software depends largely on the quality of the process used to produce it; and (2) The software process can be defined, controlled, monitored, and improved.

However, developing software is not a simple task, even with a well-defined development process. Moreover, software management, development, and maintenance have evolved from being centralized at one location to being distributed across multiple locations (Sengupta et al., 2006).

ASD approaches are often presented as an alternative to the conventional, plan-driven approach to software development (Dybå & Dingsøyr, 2008), with many claimed and argued benefits. For example, ASD approaches are supposed to improve software quality (Li et al., 2010), enhance communication (Pikkarainen et al., 2008), foster collaboration (Strode et al., 2012), and increase productivity (Sutherland et al., 2008). However, there are also some drawbacks, such as the possible loss of knowledge management due to less documentation (Stettina & Heijstek, 2011), or the potential stress on software professionals due to the constant delivery of results (Stray et al., 2012; Strode et al., 2012). Several researchers have called for more empirical studies on the effects of ASD approaches, especially on what outcomes can be expected in a large-scale, industrial context (Breivold et al., 2010; Dybå & Dingsøyr, 2008).

Banking software has some unique features, such as the need for frequent updates to keep up with the country's macroeconomic decisions. Moreover, banking software requires multiple versions and sub-versions, which makes traditional software development inefficient.

The interest in applying ASD to software development projects is driven by the corporate demand for faster and smaller delivery (Chapman & White, 2016; Chapman et al., 2017). Researchers have started to investigate ASD in software development (Gary et al., 2011). Several case studies and experience reports have been published in the academic literature on this transition in different domains, such as railway (Jonsson et al., 2012), medical science (McHugh et al., 2013), and avionics (Abrahamsson et al., 2006; Chenu, 2012). Other software domains have reported on adapting and modifying methods and practices to meet local needs (Conboy, 2009; Fitzgerald et al., 2006; Wang & Wagner, 2016). However, there are still many unanswered questions about how to choose the best adaptations and how effective they are in different situations.

Therefore, the research problem is how to apply ASD methods and practices in BSDCs and it is motivated by the need for adaptability, flexibility, and customer satisfaction in the dynamic and competitive market of banking software. The research problem is also driven by the lack of empirical evidence on the impact of ASD on the performance of BSDCs, which is a relatively under-researched domain. Addressing this research problem is expected to contribute to the literature on ASD by providing empirical evidence, insights, and recommendations for practitioners and researchers who are interested in applying or studying ASD in similar contexts.

This study explores how agile methodologies and practices influence the performance of BSDCs, using a case study of a famous information technology company in Iran[2] that has been operating in the banking industry for ten years. The study is based on semi-structured interviews with senior managers at this company, and addresses three research questions: (1) What are the benefits and drawbacks of using agile methodologies and practices in BSDCs? (2) What are the challenges in using agile methodologies and practices in BSDCs? (3) What agile methods and practices are used in BSDCs?

The main contributions of this study are: (1) It provides empirical evidence on the impact of agile methodologies and practices on the performance of BSDCs, which is a relatively under-researched domain. (2) It identifies the specific agile methods and practices that are used by the studied company, and how they are integrated into the existing process. (3) It discusses the challenges and opportunities that the studied company faced while adopting agile methodologies and practices, and how they overcame or exploited them. The main implications of this study are: (1) It suggests that agile methodologies and practices can improve the quality, communication, collaboration, and productivity of BSDCs, but also pose some risks, such as loss of knowledge management and stress on software professionals. (2) It indicates that agile methodologies and practices need to be adapted and tailored to fit the context and needs of each project and organization and that there is no one-size-fits-all solution. (3) It highlights the importance of understanding the situational context and the interactions between software processes and their environments, and how they can be improved accordingly.

2. Background

This section provides an overview of the ASD process and the characteristics of software development work for system engineering projects. It also reviews some relevant work on applying agile methodologies to the software development process.

2.1. Agile Software Development

ASD emerged in late 1990 as a response to the limitations of traditional plan-based software development methods, such as Waterfall (Bennington, 1983; Vijayasarathy & Butler, 2015; Wang et al., 2012) and the Rational Unified Process (Tanveer, 2015), which could not cope with the highly dynamic nature of software development project requirements. A common criticism of these methods was that they delivered software too slowly compared to the rate of change in the problem domain. ASD emphasizes that individuals and interactions are more important than systems and tools

Kupiainen et al. (2015) conducted a comprehensive analysis of the use and impact of software metrics in industrial ASD. Metrics are used for sprint planning, monitoring progress, improving software quality, fixing software processes, and motivating employees. In ASD, metrics such as velocity and 'effort estimate' are widely used, but companies also use their metrics to measure business value, defect count, and customer satisfaction. Table 1 summarizes the Agile Manifesto's core ideas and the features of ASD into 12 key principles (Beck et al., 2001).

[2] The case of the current study is Datis Arian Qeshm Co (Dotin), available at https://www.dotin.ir

Table 1. The Agile Manifesto's principles

#	Principles
1	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for a shorter timescale.
4	Businesspeople and developers must work together daily through the project.
5	Build projects around motivated individuals. Give them the environment and support they need and trust them to do the job.
6	A face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.
7	Working software is the primary measure of progress.
8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9	Continuous attention to technical excellence and good design enhances agility.
10	Simplicity—the art of maximizing the amount of work not done—is essential.
11	The best architectures, requirements, and designs emerge from self-organizing teams.
12	The team regularly reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

2.2. *Related Work*

ASD approaches have been predominant in software development since the mid-1990s. There are various methods of ASD, including Crystal, Dynamic systems development methodology (DSDM), XP, Feature-driven development (FDD), Kanban, Large-scale Scrum (LeSS), Lean software development (LSD), Nexus, Rapid application development (RAD), Scaled agile framework (SAFe®), Scrum, and Scrumban. As indicated in the literature, Scrum and XP stand out as the most frequently employed methods in financial institutions. Additionally, LSD, FDD, and Kanban are three other methodologies gaining popularity (Kilu, 2018). Each method possesses distinct processes and characteristics that set them apart from one another. Table 2 provides a comparison of four primary Agile Software Development (ASD) methods, considering various factors.

Table 2. Comparative analysis of four key ASDs (Nawaz et al., 2021)

Factors	Scrum Methodology	Extreme Programming	Feature-Driven Development	Kanban Approach
Design Standards	Use complex design principles	Use simple design and coding standards	Use simple design approaches	Guaranteed to reduce the waste by limiting the work in progress
Roles Description	Roles are predefined	Roles are not predefined	Roles are predefined	Roles are predefined
Complexity of Design	Design complexity is high	Low design complexity	Low design complexity	Simple design
Workflow Technique	Work in iterations. Sprints are produced.	Does not work in iteration rather follows the task flow approach	It is an incremental and iterative approach. A set of features is delivered.	Works in small iterations
Technique for Requirements Management	Product and sprint backlogs are used for managing requirements in terms of artifacts.	Story cards are used for requirement management.	Manage user requirements by building an object model of them.	Kanban Boards are used for requirement management.

Factors	**Scrum Methodology**	**Extreme Programming**	**Feature-Driven Development**	**Kanban Approach**
Product Delivery Approach	Sprints are delivered on a defined time.	Continuous Delivery	Continuous Delivery	Continuous Delivery
Standards used for coding	Coding standards are not defined.	Use defined coding standards.	Development practices are defined in advance.	Coding standards are not defined.
Testing techniques	No formal techniques are defined for performing testing.	Use several testing techniques for product auditing like acceptance testing.	Use standard testing techniques.	At the end of each increment or work product testing is performed thoroughly.
Changes acceptance	Changes are not acceptable in sprints.	It can accept changes at any phase of development.	It can accept the changing requirements of customers easily at all levels.	It can accept the changes at any time.
Process Owner	Scrum Master	Team Ownership	Each class is owned by a class owner who works under a chief programmer	Team ownership
Customer Involvement	The presence of a customer on-site is not essential.	On-site customer presence and interaction are compulsory.	For the early two phases of FDD, customer involvement is mandatory.	Not essential for the on-site availability of customers.
Project director	Scrum Master	Extreme programming Coach	Project Manager	Teamwork
Collaboration among Team	Cross-functional teams	Self-organized teams	Teamwork	The team consists of specific resources

However, the existing literature on ASD suggests that there is no consensus among practitioners on how to effectively implement and adapt ASD in software development organizations. Scott et al. (2021) investigate how LHV, a mid-sized bank, can enhance its ASD process. They conducted a case study at LHV, where they first identified eight improvement suggestions from a literature review and interviews. Then, they reported on how LHV implemented the suggestions and their perceived impact. Their results emphasize the importance of taking a consistent approach to improving agile processes by considering both business units and operations involved in the product lifecycle.

Al-Zewairi et al. (2017) reviewed studies published over 15 years (2000–2015) on agile methodologies. They found that many research studies have been conducted and many prominent approaches have been proposed in this field. They used an intuitive research approach called "Compare and Review" (CR) to perform a literature survey of the surveys of the various agile methods from January 2000 to December 2015. In 2020, Núñez et al. (2020) published another experience report. They used data from software projects to conduct a realistic experiment in a software development company. They presented and compared the results in two development scenarios: one with a traditional reactive security strategy and another with an innovative and proactive approach that applied security by default throughout the software life cycle.

Ram et al. (2019) aimed to identify success criteria for operationalizing measurements in ASD, especially factors that could facilitate the long-term use of metrics. They conducted a multiple case study where they applied ASD to operationalize process metrics at two software-intensive organizations. Hybrid methods of traditional and agile development combine the inter-team overview and predictability of long-term planning with the flexibility and agility of agile development at the team level. However, the reasons for the frequent failure of such hybrids are unclear. Bick et al. (2017) performed a case study within a large software development unit of 13 teams at a global enterprise software company to investigate how and why a combination of traditional planning at the inter-team level and agile development at the team level can lead to ineffective coordination. Based on multiple data sources, including interviews with scrum masters, product owners, architects, and senior management, and using Grounded Theory data analysis methods, Bick et al. found a lack of dependency awareness across development teams as a major cause for inadequate coordination.

The Software Development Life Cycle (SDLC) is a process that aims to produce high-quality, reliable, cost-effective, and timely software products. Various SDLC process models can be adopted for different purposes. Shylesh (2017) compared and contrasted different SDLC models based on their suitability for different scenarios. The main objective of his study was to provide an overview of some of the most popular SDLC models, such as Waterfall, Iterative, Spiral, V-Model, Big Bang, Agile, Rapid Application Development Model, and Software Prototype.

O'Connor et al. (2016) reported the results of a case study involving process discovery in a small but growing and successful software development company. Lagerberg et al. (2013) surveyed the participants of existing research projects. The study aimed to provide empirical evidence on the impact of applying agile principles and practices to large-scale industrial software development. The effective implementation of Scrum methods relies heavily on close collaboration among project stakeholders. The dispersion of project stakeholders in Global Software Development (GSD) creates significant challenges for project collaboration processes, which may limit the adoption of Scrum. Hossain et al. (2011) conducted a multi-case study examining how key project environment factors affect the application of Scrum principles in GSD. This study may be useful for researchers and practitioners who are interested in using Scrum in GSD to improve project performance.

Korkala and Abrahamsson (2007) performed two case studies on distributed ASD and compared their findings to previous communication recommendations in distributed agile development. Based on their findings and previous research, they suggest that the recommendations are worth considering in remote agile development, but with caution. Their empirical evidence shows that the role of a well-defined customer is the most important recommendation. Even in small-scale distributed ASD projects, the lack of a well-defined customer who can fulfill the responsibilities, changing requirements, and inadequate communication may create serious problems. Prikladnicki et al. (2006) published another case study research. This research aims to propose a comprehensive reference model for software development based on the findings of a case study conducted in two software development units from multinational corporations in Brazil. In 2004, a preliminary description of this model was released. This article elaborates on the reference model, strengthening its definition and examining the factors that enable multinational companies to operate effectively across geographical and cultural boundaries.

The introduction of different software development approaches requires comparing and evaluating their effectiveness. One way to conduct such a comparison is to have different teams use different process models to implement multiple versions of the same requirements. Germain and Robillard (2005) describe a novel categorization method for cognitive activities used to monitor the effort invested by six student teams while they developed simultaneous implementations of the same software requirements specifications. Three teams used a method based on the Rational Unified Process (RUP), derived from the Unified Process for Education (UPEDU). The other three teams followed a method based on the principles of the Extreme Programming (XP) methodology. However, for all teams participating, the relative importance of a category of activities that defined "active" behavior was almost constant, possibly revealing a basic behavior pattern. As a side note, aggregate differences by process model are often small and limited to a few activities, while coding-related activities dominate the effort distribution across all teams.

ASD solutions are designed to improve project work. Salo and Abrahamsson (2005) discussed the high value of validated Software Process Improvement (SPI) knowledge developed by project teams and its application at the organizational level. They suggested that the specific SPI methods of agile project teams require changes in executive-level operations for the two to coexist beneficially. The analysis highlights the need for tight coordination between the corporate and project levels throughout the projects. It lists several managerial tasks that must be performed to improve SPI in agile projects and organizations.

3. Methodology

This study adopted a case study approach to address the research problem of describing software development projects using Scrum principles in real-world situations. Case studies are suitable when the research questions are 'how' or 'why', the researcher has little control over the events, and the phenomenon is contemporary and situated in a real-life context.

The case study was conducted at a famous information technology company in the banking industry in Iran that provides various products and services to several banks. The company had started experimenting with elements of the Scrum methodology and other agile methods, and the research aimed to explore and understand the use of ASD from the practitioners' perspective.

The data collection involved five semi-structured interviews, each lasting 1-2 hours, with different company personnel involved in the transition process. The interviews were conducted by one of the researchers and were done face-to-face in an industrial setting. The interviews were complemented by informal conversations, observations, tool demonstrations, and documentation. The interview instrument (Siddique & Hussein, 2014) was developed using Wengraf's standards, as recommended by McHugh et al. (2013) (shown in Table 3).

The first step of the data collection was an unstructured interview with two senior company employees, who were the team leader of the organization's planning department and the head of PMO. This interview was exploratory and lasted 90 minutes. A memo was written to summarize the answers to the questions asked, and one of the participants confirmed the memo during a follow-up conversation. The answers to this initial interview were used to help scope the next step of the data collection.

Table 3. The process of constructing research questions follows Wengraf's method (Wengraf, 2001)

Research Purpose	Central Research Questions	Theory Questions	Example Interview Questions
Explore and understand the application of Agile Software Development from practitioners' perspectives	1. What are the opportunities for using agile methods and practices in the context of software development?	1. What benefits did they expect from Agile Software Development?	Did the ASD software affect capacity management?
		2. What benefits did they achieve or fail to achieve?	Did the ASD software affect product delivery?
	2. What are the challenges of using agile methods and practices in the context of software development?	3. What are the potential conflicts between ASD and regulatory standards?	How often were the requirements reviewed?
		4. What are the potential conflicts between ASD and organization members?	How the process was communicated to the team members?
	3. What agile methods and practices are being used in software development?	5. What agile methods and practices do they use?	Were multiple releases delivered to the customer during a project?
		6. What customizations have they made to the methods and practices they use?	How were the requirements managed during the elaboration?

Starting with a Research Purpose (RP), in this case, "to explore and understand the use of ASD from practitioners' perspective", is a top-down approach. The RP is then developed into one or more Central Research Question(s) (CRQ) that capture the main aspects of the research purpose. The RP is translated into three research questions in the current study, as described in the introduction and shown in Table 3 for completeness. Each CRQ is broken down into multiple Theory Questions (TQ), which are specific statements that will be tested during the research. We provide interview questions to help participants answer each theory question.

After preparing an initial version of the interview instrument, it was validated by an independent academic expert who was not involved in the study. The validator was contacted via email to arrange a teleconference to review all the questions in the interview instrument. To make the interview process smoother, the validator suggested rearranging the order of the questions but not the content of any of them. To familiarize the researchers with the format of the interview instrument and to estimate the timing and duration of the interviews, a series of mock interviews were conducted with non-participants in the study.

Five practitioners (Participants P1-P5) with different experiences, skills, and roles were interviewed. They had various roles such as project manager, requirements engineer, PMO member, and integration team member. P5, the fifth participant, was the head of the planning team. The first four interviewees were involved in three different projects. Some of the company's teams had previous experience with ASD in their projects. On the other hand, some were considering using it because they wanted to release it more frequently. In both cases, the interviewees had used an agile method and related practices in previous projects at the organization.

Table 4. Demographic information about the research participants

Parameters	**Participant 1**	**Participant 2**	**Participant 3**	**Participant 4**	**Participant 15**
Total work experience (Years)	11	13	12	18	14
Work experience at the studied company (Years)	4	3	3	12	10
Company Designation	PMO Manager	Head of the Planning Department	PMO staff	Chief Planning and Development Officer	Software Development Department Assistant
Academic Degree	MSc	PhD	Bachelor	MSc	MSc
Academic field	Industrial Engineer	Information Technology Management	Computer Engineer	Computer Engineer	Computer Engineer
Interview length (Minutes)	90	90	100	80	90

Participants were allowed to add to or clarify their answers after the transcribed interviews and returned them for validation. The transcripts were used for analysis after obtaining verbal consent from the participants. The interview transcripts were then analyzed to find answers to the theory questions. Wengraf (2001) principles were also used to analyze the obtained data, using a bottom-up approach to answer the questions at each level. The descriptive answers to each CRQ were examined individually, and the points raised were noted. The notes were then compared during a meeting to identify the patterns revealed. During the data analysis, it was found that there were both barriers and opportunities. Finally, the interview findings were compared with those from the literature. Besides the interview input, several KPIs were obtained to better comprehend the ASD application, as shown in Table 5. The values of these indicators were determined for each phase.

Table 5. KPIs For comparison of phases

KPI Number	**Description**
#1	Number of Employees
#2	Capacity[3]
#3	Company capital
#4	Sales (operating income)

4. Findings

4.1. Overview of Software Development at the Company

The interview responses describe the organization and approach of software development. The section begins with a description of the typical team organization for a software development project. Then, it discusses how ASD has been applied within specific sub-teams.

Each product has its team within the organization. Before adopting the ASD method, each team was managed according to the product manager's approach, which made communication difficult between the teams. This problem arose when these teams had to collaborate and deliver services. The number of team members usually ranges from 10 to 20. A Software Development Team (SDT) within a project generally consists of 5 to 15 people, with the rest of the project team focusing on other components or functions. The SDT is organized in a certain way. A small management unit consisting of a product manager, product owner, and technical leader oversees the whole team. The product manager and owner share technical and managerial responsibilities for the entire

[3] Total working hours available in the organization

product. These include the requirements, definition, design, software implementation, quality assurance, certification, and delivery phases of the software life cycle. Coordination with other product managers is also a responsibility of the product manager. The product owner is also in charge of delegating tasks to product sub-teams. The product manager handles project planning and resource allocation within the SDT. An SDT is usually organized into multiple sub-teams, each specializing in different functional elements of the project and consisting of four or five people.

4.1.1. Development Process

All of the company's products are designed to last for many years. They are divided into multiple sub-products, each intended to provide additional capabilities to the product, as agreed with the buyer(s). Delivering the product to the customer was divided into several phases. The duration of a phase varies depending on the scope of the project. A phase may last between four and six weeks in some projects and between one and six months in others. Each phase is assigned to several criteria that the project's customer must execute. After completing each phase successfully, the customer gets a delivery that contains (ideally) all of the criteria that were initially agreed on.

Within each sub-team, the company provides some flexibility in software development processes, for example, by allowing some sub-teams to use a Waterfall software development process with a single phase and others to use the Scrum methodology. As a result, one participant (P2) called their software process "*water-scrum-fall*" because Scrum was integrated into the overall project life cycle of the organization. After a phase, many software functionalities are bundled into a single integrated software release. The integration team sends software to the project customer to create a general delivery release.

According to one of the interviewees (P2), there was no systematic process for obtaining and providing customer requirements. Any customer with more bargaining power received better service. According to P2, "[…] *obtaining and providing customer requirements was not systematic. Then, powerful customers were receiving better service*."

4.1.2. Project Customers

Each customer was assigned a customer manager who received and communicated his or her requirements to the product manager. Customer and product managers communicated ad hoc, and most of their time was spent negotiating.

Participant #3 highlighted the intricate nature of stakeholder relationships in the context of this project. According to P3, '*This is a highly complex stakeholder relationship in terms of many people with different views and opinions*.' This perspective sheds light on the challenges posed by the diversity of views and opinions within the stakeholder community, emphasizing the need for effective communication and collaboration strategies to navigate this complexity. Furthermore, Participant #4 underscored the need for effective communication and collaboration with project customers. According to Participant #4, "*In our experience at the company, effective communication, and collaboration with project customers are the cornerstones of successful project outcomes. The iterative nature of our development process, especially with the implementation of a seasonal release cycle, demands continuous engagement with customers. It is not merely about delivering a product but understanding the evolving needs of the banking industry*."

Then, the customer's delivery schedule is the main factor affecting the entire schedule. Previously, the whole project management team determined the project schedule, but now the software team also inputs tasks and schedules. While the project management team sets critical milestones with the external customer, the software teams work within these parameters. This instills a sense of ownership and accountability in the team members. External customers are then informed of the agreed-upon delivery dates. Typically, if there is a risk of missing the delivery deadline, the software team will increase its size.

However, if a schedule change is necessary, it is made after consultation with the external customer. The product manager is ultimately responsible for schedule changes. Fig. 1 illustrates the relationship between the customer, the customer manager, and the product manager. Customer managers have been in direct contact with all product managers of products the customer has purchased, wasting a significant amount of time in inter-organizational coordination.

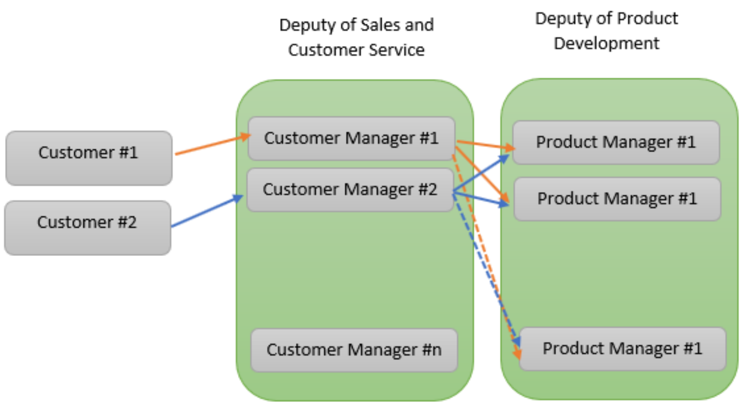

Fig. 1. The relationship between the customer, the customer manager, and the product manager

4.2. *Use of Agile Software Development*

As mentioned in the previous section and according to the interview output, various difficulties were encountered during the previous software development phase. The agile methodology was used to overcome these issues. According to Participant #4, "*The decision to adopt agile methodology at the company stemmed from our recognition of challenges in the previous software development approach. Agile offered us a more adaptive and collaborative framework, enabling us to respond swiftly to changing requirements and enhance communication within our teams. It was not just a methodology shift; it was a cultural transformation. Agile provided the flexibility and transparency we needed to navigate the complexities of our projects*." Several reasons for this change were stated in the interviews:

- The need to shorten delivery timeframes and provide the customer with incremental releases: One participant (P5) said they wanted to be "[...] *providing the customer with several more releases*." Another participant (P3), who had no prior experience with Agile Software Development, stressed the importance of frequent delivery "[...] *we would be able to serve the customer with more frequent software deployments*."

- Improving communication among the software development team: According to one of the respondents (P1), "[...] *we want more visibility in the project, i.e., who is doing what? What percentage of tasks have been completed? Among other things are cost estimates, performance, and a finished work record*." Jira Kanban boards were mentioned as being helpful in this respect.

- Increasing team member participation in the software project's coordination: Choosing one's tasks has instilled a sense of responsibility among team members. In describing the benefits of Scrum, a participant (P1) said, "*The level of involvement of some engineers is much higher* [...] *that is a significant change, and teams are working more effectively*."

- Early identification of problems: According to interviews, defects were often detected late in the integration process, requiring a more costly redo. While discussing the benefits of Agile Software Development, one member (P1) observed, "[...] *not letting things get too far before realizing something is wrong. It is all about visibility. It's about becoming aware of problems sooner*". Another participant (P4), who had no experience with Agile Software Development, said during a discussion about the benefits of using Agile in other projects, "[...] *so we get feedback sooner*."

- Allocate time for system improvement: This streamlines the flow of information between developers and customers.

4.2.1. *Executive Procedure Summary*

The production cycle process in the company is a seasonal and annual order function. The production teams have to meet the customer's demands on a seasonal basis and complete and deliver the products within the next three months. Banks and credit financial institutions are regular customers, and requests usually involve quick changes to align with the country's economic structure. Sprints for all teams start and end simultaneously, and sprints are three weeks long.

Accordingly, the production cycle is divided into three general phases:

1. Seasonal planning phase: This phase begins at the start of the last month of the season, and by the end of the season, the company prepares the seasonal production plan, which the customers approve in the first week of the new season.
2. Production phase: From the beginning of the season, the production teams analyze, produce, test, and version the customer requests specified in the seasonal production plan according to the internal production process and in regular sprints based on their priority.
3. Delivery phase: The last stage of the production process, including version testing, integration testing, and delivery to the customer in the operating environment, is called the delivery phase.

Each team is led by a scrum master responsible for activity coordination. The project is scheduled in three-week sprints with related targeted releases. At the beginning of each sprint, the team prepares a plan and tracks progress through Jira. Then, items from the backlog are prioritized for completion and assigned to the sprint.

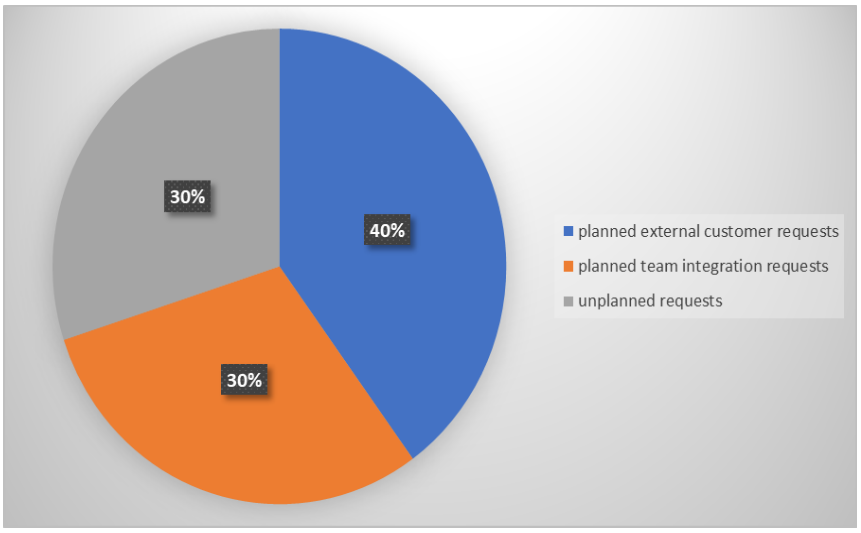

Fig. 2. Assigned capacity

4.2.2. Seasonal Planning Phase

The senior Sales and Customer Service Officer notifies the planning and monitoring unit of the customer's share of overall production capacity. In other words, this notification occurs before the planning period based on contracts for the following season. Seasonal planning lasts about five weeks. Seasonal production planning begins four weeks before the end of the season. The customer approves the seasonal production plan during the new season's first week. This phase results in a customer-approved Release Plan.

According to the annual calendar submitted by the planning and monitoring unit, customer managers undertake prioritizing actions with customer support within seven to nine days after the season's end. Support teams and customer managers complete and summarize customer requests. If necessary, a meeting with the customer is held to extract the final list of requests. The customer manager prioritizes and enters the list in Jira. The customer manager is ultimately responsible for prioritizing customer requests within the calendar month. Prioritization is the process of registering and ranking customer requests in Jira. After getting the priorities, the product owner estimates customer requests, identifies integration requests, and logs them in Jira. Customer request hours for analysis, production, and testing are assessed separately.

The planning and monitoring unit prepares integration meetings and informs product owners. The product manager, product owner, technical manager, scrum master, and planning and monitoring unit representative must attend the integration meetings. After the product manager approves the seasonal production plan, the product development deputy approves it. Customers' requests for demos are determined, in his opinion.

Customer requests are received in two ways:

1. Planned requests: Requests received during or before the planning phase and registered in Jira will form the basis of the seasonal planning of the product development teams and will be delivered in the next or later seasons.
2. Unplanned requests: Requests received during the season and the production phase (such as new Central Bank rules or requests for bug fixes) are considered unplanned requests and a certain amount of team capacity is allocated.

In each production cycle (and consequently in each sprint), 40% of the team's production capacity is considered for planned external customer requests, 30% for planned team integration requests, and 30% for responding to unplanned requests, including internal integration and customer requests. The assigned capacity is shown in Fig. 2.

4.2.3. Production Phase

The production phase is done in the form of production sprints. The teams' sprints start from the beginning of each season and simultaneously. Starting or ending sprints are made by considering the end-of-week holidays in consultation with the Vice President for Product Development and the Planning and Supervision Manager. The sprint time of all teams will be three weeks, so there is enough time to analyze, produce, test, and copy each request.

At the end of each sprint, the sprint version should be created and tested, and the first week of the sprint should be dedicated to testing its integration with the release version.

Demo sessions can be divided according to the demo audience and provider. The product owner must keep the status of requests up to date so that requests that can be demoed can be identified and reported after the sprint is completed. The product owner must keep the request's status up to date in Jira during the season. At the end of each sprint, the product owner prepares and submits requests to the deployment engineer.

4.2.4. Delivery Phase

The delivery phase is a set of actions during which the prepared version or changes are applied to the operating environment. Each team in each release cycle processes all customer requests ready to be delivered according to the stated procedure. Upon receipt of the final confirmation from the customer after delivery, the requests will be closed. Before sending the version to the deployment engineer, the product owner receives the final approval of the installation and operation from the product manager to prevent the publication of the version or change part of it if necessary. The deployment engineer delivers the output of the production phase to the operations team after completing the relevant tests and, in coordination with the customer manager, is set up in the customer's operating environment.

At the end of each season, the planning and monitoring unit prepares a report on the unit's performance in the product development process. It is communicated to the senior managers and the CEO to improve the company's operation.

5. DISCUSSION

The main purpose of this study was to explore the use of agile software development (ASD) in a large-scale software company and to identify the benefits and challenges of this approach. The study used a qualitative case study method, involving semi-structured interviews with 12 participants from different roles and levels in the company. The study also collected and analyzed KPIs and other documents related to the software development process. The findings of the study revealed that the company adopted ASD to cope with the dynamic and complex nature of the software market, to improve communication and collaboration among the software teams and with the customers, to increase the quality and frequency of software deliveries, and to enhance the satisfaction and motivation of the software developers.

The findings of this study contribute to the existing literature on ASD in several ways. First, the study provides an in-depth and comprehensive description of the software development process in a large-scale software company, covering the aspects of team organization, project planning, production, delivery, and evaluation. The study also illustrates how the company adapted and customized the ASD methods to suit its specific context and needs, such as using a hybrid approach of waterfall and scrum, allocating a certain percentage of the team capacity for unplanned requests, and conducting seasonal planning and integration meetings. Second, the study confirms and extends some of the previous findings on the benefits and challenges of ASD in large-scale software projects. For example, the study supports the claims that ASD can improve the responsiveness and flexibility of the software development process, enhance the communication and collaboration among the software teams and with the customers, increase the quality and frequency of software deliveries, and foster the satisfaction and motivation of the software developers (Dybå & Dingsøyr, 2008; Strode et al., 2012).

The study also reveals some of the difficulties and obstacles that the company encountered in adopting ASD. These include the lack of a clear and systematic process for customer requirements elicitation and prioritization, the inconsistency and variability of the software development practices across different sub-teams, the resistance of some stakeholders towards the agile principles and values, and the challenges of integrating and testing the software components from different sources and platforms. These findings are consistent with some of the previous studies that reported similar challenges and issues in large-scale ASD projects (Conboy, 2009; Fitzgerald et al., 2006; Li et al., 2010).

The findings of this study also have some practical implications for software practitioners and managers who are interested in adopting or improving ASD in their organizations. Based on the findings, some suggestions and recommendations can be made to enhance the effectiveness and efficiency of ASD in large-scale software projects. It is important to establish a clear and systematic process for customer requirements elicitation and prioritization, involving the participation and collaboration of all the relevant stakeholders, such as the customer managers, the product managers, the product owners, and the customers themselves. This can help to ensure that the software development process is aligned with the customer's needs and expectations and that the most valuable and feasible features are delivered in each sprint and release.

Furthermore, it is advisable to adopt a consistent and standardized software development practice across different sub-teams, following the same agile methods, tools, and techniques. This can help to reduce the complexity and variability of the software development process and facilitate the integration and testing of the software components from different sources and platforms. Also, it is essential to foster a culture of trust and openness among the software teams and with the customers, embracing the agile principles and values of customer collaboration, team empowerment, feedback, and adaptation. This can help to overcome the resistance and skepticism of some stakeholders towards the agile approach and to enhance communication and collaboration among the software teams and with the customers.

While the study provides valuable insights into the positive impact of ASD on KPIs, it is essential to acknowledge certain limitations about the qualitative nature of data collection, which relied on interviews with heads, managers, and staff. While these interviews yielded rich perspectives and firsthand accounts, the interconnected nature of some KPIs poses a challenge in establishing a causal relationship between ASD and KPI improvements. Future research could benefit from a more objective examination of this relationship. It is recommended to conduct empirical studies that employ quantitative methods, such as controlled experiments or longitudinal analyses, to further investigate and validate the observed improvements in KPIs (such as the number of employees, the capacity, the company capital, and the sales) attributed to the ASD method. This would contribute to a more comprehensive understanding of the dynamics between ASD methodologies and key performance outcomes in various organizational settings.

5.1. Implementation Challenges

We have identified some common obstacles to implementing software process improvement methodologies based on our case study work. We describe them below and offer some suggestions to overcome them

- How to start: Some organizations faced difficulties with software process improvement and their chosen approaches. The planning and development team provided guidance and contact information to help these teams understand the process better. The planning and development team also met with the stakeholders several times to ensure they were clear about the process.

- Lack of knowledge and information: Some individuals were unable to participate in process improvement due to a lack of understanding. When employees lacked information, they could not give feedback to others during problem-solving sessions or suggest ways to complete a task. As a result, they paid little attention to reporting and resolving problems. The main reason for the lack of process improvement knowledge was a lack of familiarity with the agile methodology.

- Resistance to change: the studied company in the current research, like any other company, encountered resistance to change. Employees' resistance resulted from their lack of awareness of the upcoming process. Senior management support was crucial to overcome this resistance and accept the process change.

5.2. Implementation Opportunities

We have analyzed the opportunities that the ASD process provides as a tool for managing software development and found several important applications. Table 6 shows the selected KPIs that we used to measure the process. We have also described how we applied the ASD process in agile projects. Our findings revealed three main opportunities for using the ASD process, which we discuss in this section.

- Improving internal team relationships: Before using the process, many of our staff spent time negotiating and integrating software. By using this process and Jira, we saved a lot of time and used it to improve the software instead.
- Improving customer service: As we mentioned earlier, the customer did not receive any feedback on the status of their request after submitting it. By using this process, the customer can check the status of their request at any time.
- Increasing the satisfaction of stakeholders: The service delivery to domestic and foreign stakeholders (both internal and external) was greatly improved. Commitment to the service-level agreement: By clearly defining the tasks of each role in the process, we increased our commitment to the service-level agreement and responded to the customer promptly.

Table 6. KPIs' Values

KPI #	Description	1st Season*	2nd Season	3rd Season	4th Season	5th Season	6th Season	7th Season	8th Season
#1	Number of Employees	652	734	820	881	971	1050	1108	1150
#2	Capacity (Hour)	NA	49797	73315	73814	69310	84590	100536	104710
#3	Company Capital (Billion Iranian Rial)	1000	1000	1380	1380	2500	2500	2500	2500
#4	Sales (Billion Iranian Rial)	545	1141	1899	3354	17	1293	3974	5000

*. Each season spans a standard three-month period.

5.3. Recommendations for Practice

According to one of the interviewees (P3), one of the issues that took a long time was the coordination of the Deputy of Sales and Customer Service. This was a waste of time because this office was not informed from the start of the design process. Therefore, we recommend that in similar situations, all stakeholders in the process should be informed about the problems and task definitions from the beginning, rather than relying only on the support of the top management of the organization. Another issue is the employees' understanding of whose roles and responsibilities are changing. One of the challenges of implementing the process was the resistance to accepting changes in organizational roles. For this reason, we recommend that the tasks of each part should be partially extracted at the outset and communicated to the staff

6. Conclusion

This study explored how to apply ASD methodologies to the development of banking systems, based on the views and experiences of a panel of experts from various backgrounds and roles. The study found that heterogeneous software development teams faced challenges in adopting ASD and navigating multiple options. The study also discovered that cultural resistance within the organization temporarily hindered the implementation of ASD approaches. Interviews were the most effective data collection method, but they were complemented by informal conversations, observations, equipment demonstrations, and documentation. This article examined the difficulties and benefits of using agile methodology in organizations that design and produce banking systems. The results of this study were derived from the perspectives of the internal experts and the values of the indicators.

We could not access project information due to the confidential nature of most of the company's activities. Our findings largely depended on the opinions of the people who participated in our interviews. This resulted in our inability to verify them by examining other sources of information, such as project software repositories or software process documentation. We selected who to interview based on their job title, their availability, and the diversity of their opinions and experiences with Agile Software Development.

Despite the limitations mentioned, the study identifies some important themes related to ASD processes and provides a roadmap for addressing the associated challenges and opportunities. In addition to these general issues, we have proposed a series of urgent research topics to guide future work in this field. In future studies, the cost and time saving of agile methodologies compared to traditional approaches can be quantified. In addition, the impact of using an ASD methodology on customer satisfaction and product quality at different stages can be measured.

REFERENCES

Abrahamsson, P., Marchesi, M., & Succi, G. (2006). *Extreme Programming and Agile Processes in Software Engineering: 7th International Conference, XP 2006, Oulu, Finland, June 17-22, 2006, Proceedings* (Vol. 4044). Springer.

Al-Zewairi, M., Biltawi, M., Etaiwi, W., & Shaout, A. (2017). Agile software development methodologies: survey of surveys. *Journal of Computer and Communications*, *5*(05), 74.

Andres, C. (2005). *Extreme programming explained: embrace change*. Addison-Wesley.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., & Jeffries, R. (2001). Manifesto for agile software development.

Benington, H. D. (1983). Production of large computer programs. *Annals of the History of Computing*, *5*(4), 350-361.

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. (2017). Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. *IEEE Transactions on Software Engineering*, *44*(10), 932-950.

Black, S., Boca, P. P., Bowen, J. P., Gorman, J., & Hinchey, M. (2009). Formal versus agile: Survival of the fittest. *Computer*, *42*(9), 37-45.

Boehm, B. (2002). Get ready for agile methods, with care. *Computer*, *35*(1), 64-69.

Boehm, B. W., Boehm, B., & Turner, R. (2004). *Balancing agility and discipline: A guide for the perplexed*. Addison-Wesley Professional.

Breivold, H. P., Sundmark, D., Wallin, P., & Larsson, S. (2010). What does research say about agile and architecture? 2010 Fifth International Conference on Software Engineering Advances,

Chapman, R., & White, N. (2016). Industrial experience with agile in high-integrity software development. Developing Safe Systems: Proc. 24th Safety-Critical Systems Symp., Brighton, UK,

Chapman, R., White, N., & Woodcock, J. (2017). What can agile methods bring to high-integrity software development? *Communications of the ACM*, *60*(10), 38-41.

Chenu, E. (2012). Agile & Lean software development for avionic software. Embedded Real-Time Software and Systems (ERTS2012),

Clarke, P., & O'Connor, R. V. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and software technology*, *54*(5), 433-447.

Clarke, P., O'Connor, R. V., Leavy, B., & Yilmaz, M. (2015). Exploring the relationship between software process adaptive capability and organisational performance. *IEEE Transactions on Software Engineering*, *41*(12), 1169-1183.

Coleman, G., & O'Connor, R. (2008). Investigating software process in practice: A grounded theory perspective. *Journal of Systems and Software*, *81*(5), 772-784.

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information systems research*, *20*(3), 329-354.

Curtis, B. (1989). Modeling the software process: three problems overcome with behavioral models of the software development process (panel session). Proceedings of the 11th International Conference on Software Engineering,

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, *50*(9-10), 833-859.

Felsing, J. M., & Palmer, S. R. (2002). A practical guide to feature-driven development. *IEEE Software*, *7*, 67-72.

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems*, *15*(2), 200-213.

Fruhling, A., & Vreede, G.-J. D. (2006). Field experiences with eXtreme programming: developing an emergency response system. *Journal of Management Information Systems*, *22*(4), 39-68.

Gary, K., Enquobahrie, A., Ibanez, L., Cheng, P., Yaniv, Z., Cleary, K., Kokoori, S., Muffih, B., & Heidenreich, J. (2011). Agile methods for open source safety-critical software. *Software: Practice and Experience*, *41*(9), 945-962.

Germain, É., & Robillard, P. N. (2005). Engineering-based processes and agile methodologies for software development: a comparative case study. *Journal of Systems and Software*, *75*(1-2), 17-27.

Glas, M., & Ziemer, S. (2009). Challenges for agile development of large systems in the aviation industry. Proceedings of the 24th ACM SIGPLAN conference companion on Object-oriented programming systems languages and applications,

Hohl, P., Klünder, J., van Bennekum, A., Lockard, R., Gifford, J., Münch, J., Stupperich, M., & Schneider, K. (2018). Back to the future: origins and directions of the "Agile Manifesto"–views of the originators. *Journal of Software Engineering Research and Development*, *6*(1), 1-27.

Hossain, E., Bannerman, P. L., & Jeffery, R. (2011). Towards an understanding of tailoring scrum in global software development: a multi-case study. Proceedings of the 2011 International Conference on Software and Systems Process,

Jonsson, H., Larsson, S., & Punnekkat, S. (2012). Agile practices in regulated railway software development. 2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops,

Kilu, E. (2018). *Software Process Improvement Using Agile Methods in Financial Institutions. LHV Bank Case* [Master's Thesis, University of Tartu]. Tartu. https://core.ac.uk/download/pdf/237084352.pdf

Korkala, M., & Abrahamsson, P. (2007). Communication in distributed agile development: A case study. 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007),

Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2015). Using metrics in Agile and Lean Software Development–A systematic literature review of industrial studies. *Information and software technology*, *62*, 143-163.

Lagerberg, L., Skude, T., Emanuelsson, P., Sandahl, K., & Ståhl, D. (2013). The impact of agile principles and practices on large-scale software development projects. *Linkopings university, SE-581*, *83*.

Li, J., Moe, N. B., & Dybå, T. (2010). Transition from a plan-driven process to scrum: a longitudinal case study on software quality. Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement,

McHugh, M., Cawley, O., McCaffcry, F., Richardson, I., & Wang, X. (2013). An agile v-model for medical device software development to overcome the challenges with plan-driven software development lifecycles. 2013 5th International Workshop on Software Engineering in Health Care (SEHC),

McLeod, L., & MacDonell, S. G. (2011). Factors that affect software systems development project outcomes: A survey of research. *ACM Computing Surveys (CSUR)*, *43*(4), 1-56.

Nawaz, M., Nazir, T., Islam, S., Masood, M., Mehmood, A., & Kanwal, S. (2021). Agile Software Development Techniques: A Survey: Agile Software Development Techniques: A Survey. *Proceedings of the Pakistan Academy of Sciences: A. Physical and Computational Sciences*, *58*(1), 17-33. https://www.paspk.org/wp-content/uploads/2021/08/ES-721.pdf

Núñez, J. C. S., Lindo, A. C., & Rodríguez, P. G. (2020). A preventive secure software development model for a software factory: a case study. *IEEE Access*, *8*, 77653-77665.

O'Connor, R. V., & Clarke, P. (2015). Software process reflexivity and business performance: initial results from an empirical study. Proceedings of the 2015 International Conference on Software and System Process,

O'Connor, R., Elger, P., & Clarke, P. M. (2016). Exploring the impact of situational context—A case study of a software development process for a microservices architecture. 2016 IEEE/ACM International Conference on Software and System Processes (ICSSP),

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and agile software development. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.,

Paige, R. F., Galloway, A., Charalambous, R., Ge, X., & Brooke, P. J. (2011). High-integrity agile processes for the development of safety-critical software. *International Journal of Critical Computer-Based Systems*, *2*(2), 181-216.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, *13*(3), 303-337.

Prikladnicki, R., Audy, J. L. N., & Evaristo, R. (2006). A reference model for global software development: findings from a case study. 2006 IEEE International Conference on Global Software Engineering (ICGSE'06),

Ram, P., Rodriguez, P., Oivo, M., & Martínez-Fernández, S. (2019). Success factors for effective process metrics operationalization in agile software development: A multiple case study. 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP),

Salo, O., & Abrahamsson, P. (2005). Integrating agile software development and software process improvement: a longitudinal case study. 2005 International Symposium on Empirical Software Engineering, 2005.,

Scott, E., Milani, F., Kilu, E., & Pfahl, D. (2021). Enhancing agile software development in the banking sector—A comprehensive case study at LHV. *Journal of software: evolution and process*, *33*(7), e2363.

Sengupta, B., Chandra, S., & Sinha, V. (2006). A research agenda for distributed software development. Proceedings of the 28th International Conference on Software Engineering,

Shylesh, S. (2017). A study of software development life cycle process models. National Conference on Reinventing Opportunities in Management, IT, and Social Sciences,

Siddique, L., & Hussein, B. A. (2014). Practical insight about choice of methodology in large complex software projects in Norway. 2014 IEEE International Technology Management Conference,

Stettina, C. J., & Heijstek, W. (2011). Necessary and neglected? An empirical study of internal documentation in agile software development teams. Proceedings of the 29th ACM International Conference on Design of Communication,

Stray, V. G., Moe, N. B., & Dybå, T. (2012). Escalation of commitment: a longitudinal case study of daily meetings. International Conference on Agile Software Development,

Strode, D. E., Huff, S. L., Hope, B., & Link, S. (2012). Coordination in co-located agile software development projects. *Journal of Systems and Software*, *85*(6), 1222-1238.

Sutherland, J., Jakobsen, C. R., & Johnson, K. (2008). Scrum and CMMI level 5: The magic potion for code warriors. Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008),

Tanveer, M. (2015). Agile for large-scale projects—A hybrid approach. 2015 National Software Engineering Conference (NSEC),

Vijayasarathy, L. R., & Butler, C. W. (2015). Choice of software development methodologies: Do organizational, project, and team characteristics matter? *IEEE Software*, *33*(5), 86-94.

Wang, X., Conboy, K., & Pikkarainen, M. (2012). Assimilation of agile practices in use. *Information Systems Journal*, *22*(6), 435-455.

Wang, Y., & Wagner, S. (2016). Towards applying a safety analysis and verification method based on STPA to agile software development. 2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED),

Wengraf, T. (2001). *Qualitative research interviewing: Biographic narrative and semi-structured methods*. sage.